НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Інформаційно-телекомунікаційних мереж

До захисту допущено:

Завідувач кафедри

_____ Лариса ГЛОБА

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
спеціальності 172 «Телекомунікації та радіотехніка»
на тему: «Методи попередньої обробки великих даних»

Виконала:
студентка IV курсу, групи ТІ-62
Гребініченко Марія Володимирівна _____

Керівник:
Асистент кафедри ІТМ ІТС
Бугаєнко Юрій Михайлович _____

Рецензент:
Доцент кафедри ТС ІТС,
Созонник Галина Дмитрівна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

ЗАВДАННЯ
на дипломну роботу студентці
Гребініченко Марії Володимирівні

1. Тема роботи «**Методи попередньої обробки великих даних**», керівник роботи Бугаєнко Юрій Михайлович, асистент кафедри інформаційно-телекомунікаційних мереж ІТС, затверджені наказом по університету від «30» березня 2020 р. № 924-с

2. Термін подання студентом роботи 8 червня 2020 р.

3. Вихідні дані до роботи теорія попередньої обробки інформації, наукові статті про методи очищення великих даних

4. Зміст роботи

1. Проаналізувати підходи щодо попередньої обробки великих даних, математичні методи, які дозволяють відтворити спотворені дані

2. Запропонувати вдосконалений метод щодо очистки даних

3. Створення прототипу системи для демонстрації та перевірки отриманих результатів

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Тема, актуальність, мета, задачі.
2. Попередня обробка великих даних
3. Модифікований метод очищення даних.
4. Порівняння запропонованого методу з уже існуючими.
5. Загальні висновки.

6. Дата видачі завдання 15 вересня 2019 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблематики та необхідності попередньої обробки великих даних	15.09.2019-02.11.2019	виконано
2	Дослідження існуючих підходів та математичних методів для очищення даних	03.11.2019-30.11.2019	виконано
3	Підготовка публікації (OSTIS 2020)	01.12.2019-07.12.2019	виконано
4	Розробка вдосконаленого методу попередньої обробки даних для підвищення їх якості	02.02.2020-29.02.2020	виконано
5	Розробка програмного прототипу запропонованого алгоритму	01.01.2020-16.04.2020	виконано
6	Отримання і аналіз результатів вдосконаленого запропонованого методу очищення	17.04.2020-16.05.2020	виконано
7	Підготовка тексту диплома	17.05.2020-01.06.2020	виконано

Студент Марія ГРЕБІНІЧЕНКО

Керівник Юрій БУГАЄНКО

ЗМІСТ

РЕФЕРАТ

Робота містить 54 сторінки, 1 таблицю та 20 рисунків. Було використано 16 джерел.

Мета роботи: підвищення якості обробки великих обсягів обчислюваних даних за рахунок очищення даних від випадково виникаючих помилок.

В ході виконання даної роботи було проаналізовано проблему зниження якості вхідних даних у системах обробки великих обсягів інформації. Розглянуто основні методи попередньої обробки наборів даних, для кожного означені основні недоліки та переваги. На базі проведеного аналізу запропоновано вдосконалений метод очистки даних за рахунок поєднання існуючих ефективних підходів у єдиний алгоритм, що динамічно аналізує дані та визначає необхідні типи очистки.

Виконана програмна реалізація запропонованого алгоритму та класичних методів очистки даних у вигляді веб-додатку.

Ключові слова: великі дані, методи попередної обробки, очищення даних

ABSTRACT

The work consists of 54 pages, 1 table and 20 figures. Used 16 references.

Goal of diploma: improving quality of big data processing based on data cleansing of random errors.

In the course of this work, the analysis of the problem of decreasing input data quality in big data processing systems was carried out. The main methods of preprocessing datasets are considered, advantages and disadvantages of each are formulated. On the base of analisys improved data cleansing method of combining existed approaches in a single algorithm is proposed, which perform data analysis and dynamically set up necessary data cleansing instruments.

Software realization (web-application) of proposed algorithm and basic data cleansing methods was developed.

Key words: big data, data preprocessing, data cleansing

ПЕРЕЛІК СКОРОЧЕНЬ

IQR	Interquartile range
DQ	Data quality metrics
BD	Big data
DS	Data science
DC	Data cleansing

ВСТУП

Актуальність. Сьогоденний світ сприяє все швидшому накопиченню великих обсягів інформації, що посприяло розвитку нової галузі науки, як Big Data – сукупність підходів та принципів роботи з великими масивами даних. Просте зберігання інформації не може напряму надати користь людству, знання несуть у собі розвиток лише при використанні їх у дії. Майже неможливо уявити обсягів даних, що зберігаються зараз у будь-яких формах у світі. Саме тому ручне опрацювання такої кількості інформації втрачає ефективність, і галузь Big Data знаходить нові рішення щодо її якісної обробки.

На вхідні дані, які пізніше піддаються обробці та аналізу, впливають ряд різноманітних чинників, як збирання та формування набору з різних джерел, передача їх через світову мережу, умови збору інформації з датчиків (температурний, механічний впливи), програмний збій та безліч інших. Кожен з цих факторів потенційно знижує якість даних, що підлягають обробці. Як наслідок у кінцевих датасетах можуть виникати невалідні, хибні, пропущені значення, що практично знеможливлює отримання високоякісних результатів обчислень великих даних.

Існує ряд методів та алгоритмів, що розв‘язують дану проблему, проте вони не є максимально ефективними , або вирішують проблему очистки даних лише частково. Саме тому розробка нових удосконалених методів на сьогоднішній день є актуальною задачею.

Об’єкт роботи: процес попередньої обробки інформації, що використовується при роботі з Big Data.

Предмет роботи: методи підвищення якості вхідних даних перед їх подальшою обробкою або аналізом

Мета роботи: підвищити достовірність результату в обробці великих обчислюваних даних за рахунок очищення даних від випадково виникаючих помилок

Для досягнення мети дослідження було поставлено та вирішено такі **основні задачі:**

1. Проаналізувати підходи щодо попередньої обробки великих даних, математичні методи, які дозволяють відтворити спотворені дані

2. Запропонувати вдосконалений метод щодо очистки даних

3. Створення прототипу системи для демонстрації та перевірки отриманих результатів

Теоретичний результат дослідження:

1. Аналіз методів попередньої обробки даних для систем, що працюють з великими обсягами даних, який показав що сучасні методи очистки не є досконалими і потребують модифікацій.

2. Вдосконалений метод очистки даних, що підвищує якість кінцевого результату обробки за рахунок поєднання проаналізованих методів попередньої обробки інформації у єдиний модифікований алгоритм та їх застосування при необхідних умовах, що визначаються на основі аналізу вхідних даних.

Практичний результат роботи:

1. На основі запропонованого алгоритму реалізовано програмне забезпечення системи очищення великих даних, яке дозволяє обробляти великі обсяги інформації та як наслідок підвищити достовірність фінального результату обробки.

2. На базі розробленого програмного забезпечення у вигляді вебсервісу проведено модельний експеримент для набору даних щодо роботи серверних машин, який доводить ефективність запропонованого рішення.

РОЗДІЛ 1.
АНАЛІЗ ПІДХОДІВ ЩОДО ПОПЕРЕДНЬОЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ

1.1. Основні поняття, пов‘язані з напрямом Big Data

Словосполучення «великі дані» вже давно не вживається виключно у прямому розумінні. На сьогоднішній день майже неможливо сказати, який саме об‘єм даних можна вважати великим. Крім того, саме по собі існування великих масивів даних не несе у собі користі: лише результат їх аналізу або використання їх у машинному навчанні має подальший вплив на кінцевий продукт та може змінювати наше подальше життя. Дійсно, у терміні Big Data мається на увазі саме великі обсяги інформації, проте окрім лише отримання та встановлення факту існування широкого набору даних, невід‘ємною частиною стала сукупність методів, програмних та апаратних засобів, що виконують подальшу обробку.

Аналітики компанії IBS підрахували, що загальний об‘єм даних у світі може бути рівним таким значенням [11]:

Таблиця 1.1.

Приблизна кількість даних у світі за роками

Рік	Кількість даних
2003	5 екзабайтів (1 ЕБ = 1 млрд ГБ)
2008	0,18 зетабайтів (1ЗБ = 1024 ЕБ)
2015	більше ніж 6,5 зетабайтів
2020	40-44 зетабайтів
2025 (прогноз)	400-440 зетабайтів

В загальному, явище Big Data – це сукупність даних в такому об‘ємі, що вимоги щодо їх обробки перевищують потужність стандартних БД, та набір інструментів та технологій, що уможливлюють роботу з ними. Великі дані мо-

жуть набувати будь-яких типів та форматів, як листування електронною поштою або за допомогою месенжерів, знімки з камер спостереження та спутників, стрімінгові аудіо- та відео дані, код веб-сторінок, дані фінансових ринків, тексти наукових або комерційних робіт, банківські транзакції та багато інших.

Існує так званий набір ознак великих даних - VVV(volume, velocity, variety), фізичний об‘єм, швидкість приросту інформації та необхідності до їх швидкої обробки, можливість роботи з даними різних типів у один і той самий час. Таке характеризування big data було розроблено міжнародною компанією Meta Group у 2001 році з метою вказати на головні аспекти роботи з великими обсягами даних у відповідних умовах.

Базуючись на вищесказаному, можливо виділити наступні основні принципи роботи з великими даними:

1. Горизонтальна масштабованість. Це - базовий принцип обробки великих даних. Загальні об‘єми даних ростуть з кожним днем і саме тому необхідно підвищувати кількість обчислювальних вузлів, за якими відбувається їх розподілення, причому обробка повинна відбуватися без погіршення продуктивності.

2. Відмовостійкість. Цей принцип випливає з попереднього. Оскільки обчислювальних вузлів може бути багато (іноді десятки тисяч) і їх кількість найвірогідніше буде збільшуватися, зростає і шанс виходу машин з ладу. Методи роботи з великими даними повинні враховувати можливість таких ситуацій і передбачати превентивні заходи.

3. Локальність даних. Так як дані розподілені по великій кількості обчислювальних вузлів, то, якщо вони фізично знаходяться на одному сервері, а обробляються на іншому, витрати на передачу даних можуть стати невиправдано великими. Тому обробку даних бажано проводити на тій же машині, на якій вони зберігаються.

1.2. Аналіз проблематики та необхідності у попередній обробці даних

На сьогоднішній день технології та науки, пов‘язані з великими обсягами даних, не є новими, продовжують розвиватись та не втрачають своєї актуальності. Питання та проблеми, що стосуються обробки та зберігання масивів даних торкаються майже всі сфери бізнесу, освіти та інфраструктури.

Більшість великих компаній сьогодення вже не задовольняються єдиним джерелом даних, таких як персональних даних працівників та клієнтів, геопросторові, текстові, відео- та аудіо- дані. Інформація, зібрана з різних джерел, можуть мати різну структуру через архітектуру бази даних, і при їх інтеграції виникає питання формування достовірної інформації у єдиному форматі та сховищі. Окрім того, велика вірогідність створення дубльованих або ненормалізованих записів.

В технічних системах, пов‘язаних з Інтернетом речей, некоректні дані можуть надходити до системи через ряд різних факторів, таких як температурні впливи, механічний або програмний збій, умови, що непередбачені для коректної роботи датчиків, перенавантаження обробляючої системи та ін. В результаті у кінцевих датасетах можуть виникати невалідні, хибні, пропущені значення.

Не менш актуальним це питання є для установ та сфер, що бажають перейти з паперової форми зберігання інформації на цифрову.

Вищенаведені випадки отримання спотвореного великого обсягу інформації є найпоширенішими, проте не єдиними. Безпосередня робота з даними такої якості та результати, що будуть отримані, не можуть претендувати на правильність та актуальність. Така інформація потребує попередньої обробки та підготовки для підвищення якості кінцевого продукту обробки за допомогою математичних алгоритмів та систем. Набори таких інструментів об‘єднують в єдине поняття, що називається очисткою даних.

Основний процес очищення складається з таких етапів [2]:

1) *Аналіз наданих даних:* на цій фазі складається набір метаданих про надані масиви інформації вручну або за допомогою програмної реалізації.

2) Визначення правил і порядку обробки даних: визначення методу, який буде застосований до заданого набору інформації, перевірка чи отримані всі необхідні супутні дані, що необхідні для його виконання (наприклад, параметри, що задаються експертом)

3) Валідація: перевірка чи очищення даних відбувається згідно сформованих метаданих, чи є обробка правильною та ефективною; проводиться ітераційно із аналізом (часто – на певній частині даних) до тих пір, доки не буде досягнуто найкращого результату;

4) Перетворення: після підтвердження, що очищення даних відбувається коректно та ефективно, виконані зміни застосовуються до всього датасету;

5) Заміщення даних: для оновлення даних у джерелі, звідки дані пізніше будуть взяті для обробки, очищені коректні дані заміняють їх у сховищі.

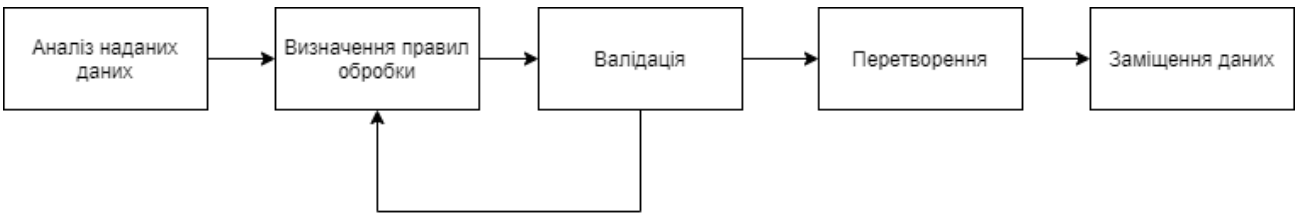

Рис. 1.1. Блок-схема базового процесу очистки даних

Точність очистки даних вимірюємо за таким виразом:

$$AC = \frac{n\,(C \cap B)}{n\,(C\,)} \qquad (1.1)$$

де С – початковий набір «забруднених» даних, В – набір даних після трансформації.

Кількісну втрату даних вимірюємо за таким виразом:

$$DL = \frac{n\,(C - B)}{n\,(C\,)} \qquad (1.2)$$

де C – початковий набір «забруднених» даних, B – набір даних після трансформації. [2]

1.3. Аналіз існуючих методів очистки даних

1.3.1. Методи очистки для даних, що збираються з декількох джерел

Найпоширенішою метою обробки великих даних є підбивання підсумків та статистики щодо роботи у багатьох філіалах компаній в різних містах, або обробка інформації, залишеною клієнтами тощо. У таких випадках інформація, що збирається, не є структурованою за єдиним шаблоном через різні архітектури сховищ даних, причини та обставини збору та ін.

Сьогодні більшість даних вже не очищується вручну, адже встановити єдиний формат для великих даних цим способом майже неможливо через високе споживання людських ресурсів та велику ймовірність помилок і неякісної обробки. Майже весь процес підвищення їх якості виконується на базі правил обробки та перетворення, що встановлюються експертами.

Проте при отриманні датасету з різних джерел, постає проблема валідації отриманої інформації, адже у кожному сховищі можуть бути встановлені різні норми для даних, а загальні правила не зможуть забезпечити їх максимальну коректність та якість. Встановлення правил для кожного джерела окремо як розв‘язок цієї задачі має право на існування, проте не є найоптимальнішим через надзвичайно високу кількість правил. Якщо збір інформації відбувається з k джерел, отриманий датасет містить в собі n атрибутів, кожен з яких у середньому містить зв‘язок з $a(n)$ параметрів, m – кількість параметрів для кожного джерела, що відрізняються і потребують додаткового встановлення правил, а $b(m)$ – кількість параметрів, щодо яких нові правила повинні бути сформовані, то загальна кількість встановлення експертами нових правил дорівнює $m*b(m)*a(n)*k$. При зростанні кількості джерел або кількості параметрів, загальна кількість модифікованих правил, що має бути встановлена різко зростає, що робить неможливим їх створення повністю експертами для даних високої різноманітності.

Один з методів очистки даних [4] вирішує цю проблему за допомогою програмного формування правил шляхом поєднання базових, що встановлені експертом, та сформованих з метаданих на етапі аналізу та визначення порядку обробки інформації. Математична модель асоціативного правила має вигляд $R(X \Rightarrow Y: C=b\%\ ,\ S=a\%)$, де X та Y виражають непорожні (диз'юнктивні) підмножини атрибутів, C – точність правила, що дорівнює відношенню кількості даних, що входять до X та Y, та кількості даних, що входять до X, S – підтримка правила, що дорівнює відношенню кількості даних, що входять до X та Y до загальної кількості даних, тобто до якого відсотку датасету це правило зможе бути застосованим. Для даної моделі вираз (Attr(Xi)) ⇒{Attr(Yj)}, Xi ∈X, Yj∈Y буде виражати базу правила. Якщо два асоціативних правила R1 та R2 мають однакову базу, то це позначається виразом $R1=_{BA}R2$.

Припустимо, що існує два асоціативних правила $r1(\{a1=v_{(r1)1},a2=v_{(r1)2}\} \Rightarrow \{a3=v_{(r1)3}\}: C=b\%\ ,\ S=a\%)$ та $r2(\{a1=v_{(r2)1},\ a2=v_{(r2)2}\} \Rightarrow \{a3=v_{(r2)3}\}: C=d\%,\ S=c\%)$, в яких $v_{(ri)j}$ це значення j-го параметра та однакова база правила. Тоді поєднання у складене правило br1 буде формуватися таким чином:

1. Якщо $v_{(r1)j} \neq v_{(r2)j}$, то $br1 = (\{a1=_{v(r1)1},a1=_{v(r2)1}\},\{a2=_{v(r1)2},\ a2=_{v(r2)2}\}) \Rightarrow (\{\{a3=_{v(r1)3},a3=_{v(r2)3}\}\}):C=((a+c)bd/(ad+bc))\ ,\ S=\ (a+c)\ \%)$,
2. Якщо $v_{(r1)j} = v_{(r2)j}$, то $br1 = (\{a1=v_{(r1)1}\},\ \{a2=v_{(r1)2}\}) \Rightarrow \{\{a3=v_{(r1)3}\}\}$: C=b%, S=a %) =r1=r2
3. $r1+_{BA}r2= r2+_{BA}r1$

Останній вираз показує, що два правила є взаємозамінні, а отже інтеграція може виконуватися на основі будь якого з двох правил. Після формування правил, вони проходять валідацію (чи підтримують дані цього джерела обробку) та програмно застосовуються для кожного джерела даних у очистці.

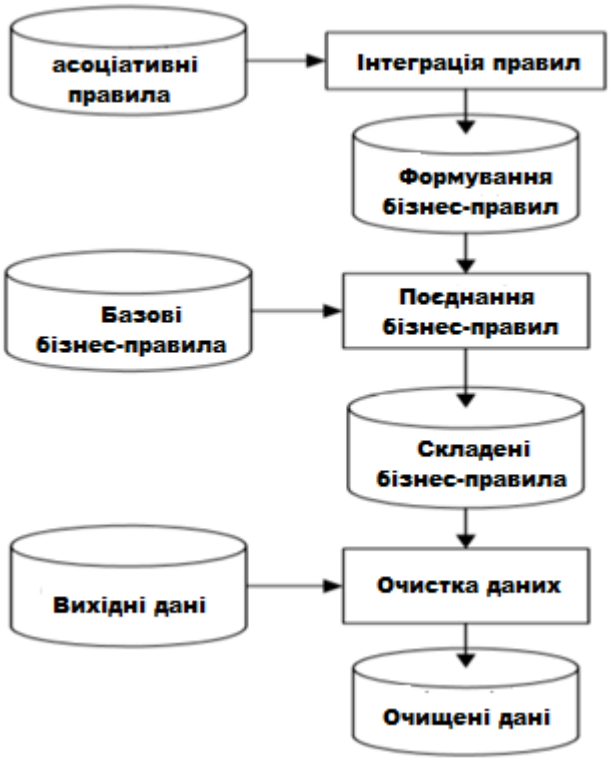

Рис. 1.2. Базовий алгоритм очистки даних із застосуванням асоціативних правил

Цей метод є оптимальним діагностичним методом: він формує правила валідації на основі метаданих, наданих експертом, визначає спотворені значення, проте не зможе виправити їх. Єдиний засіб виправлення даних, що зможе впровадити застосування даного підходу – лише видалення. При значних забрудненнях даних, втрати корисної інформації можуть бути критичними та в результаті кінцевої обробки надати експертові хибний висновок. Окрім цього, алгоритм нездатний виявити чи були надані дубльовані записи із різноманітних джерел. Повторюваність одних і тих самих значень, навіть валідних відповідно до сформованих правил, має значний вплив на кінцевий результат роботи з датасетом.

1.3.2. Пошук та відновлення відсутніх значень [5]

Майже завжди інформація, що надходить на обробку є пошкодженою, значення параметрів можуть бути невалідними, або повністю відсутніми. Причиною цьому явищу може бути широкий ряд факторів, таких як пошкодження датчиків збору інформації, проблеми з мережею або втрата пакетів при передачі даних по мережі інтернет, помилки при виконанні транзакцій у базі даних під час запису значень у пам‘ять та інші.

Помилки та проґалини у наборах інформації можуть призводити до значного зниження якості кінцевого результату обробки або до помилок у процесі обчислення та навіть неможливості отримання будь якого результату. Найочевиднішим розв‘язком проблеми може здаватись позбутись їх, проте є кілька факторів, що вказують на кращий метод – відновити їх. Видалення нерелевантних записів може призвести до значного скорочення кількості інформації для обробки, що може виявитись критичним для кінцевого результату. Крім того, можлива ситуація втрати потенційної користі даних, наприклад у такій ситуації: не були зібрані дані щодо переданого трафіку у години найвищого навантаження мережі, і відсутність пікових значень значно вплине на обробку, і фінальний результат міститиме значну похибку. Аби запобігти подібним ситуаціям, необхідні механізми для дуже точного передбачення пропущених або видалених значень.

Наразі існує велика різноманітність методів щодо роботи з відсутніми значеннями. Один з найпростіших способів – встановлення значення за замовчуванням, якщо воно вказане у метаданих або якщо тип відсутнього значення може мати його. При його відсутності є практика заповнювати втрачені дані випадковим значенням з діапазону можливих для даного типу, проте такий метод може значно вплинути на кінцеві обчислення і викликати високу похибку. Точності може принести застосування сортування за ключовим параметром та при-

своєння значень у діапазоні записів, що знаходяться поряд. Проте і цей метод не дає високої гарантії щодо високоточних результатів при роботі з цими даними.

При спробах надати випадковому значенню більшої значущості виник підхід щодо групування даних за певним показником (або групі показників), і всім пропущеним значенням копіювати показники з їх умовної групи. При застосуванні на практиці, може відбуватись досить неоднозначна ситуація, особливо при великих кількостях пошкоджених даних – усі записи з відсутніми значеннями групуються разом, і нічого більше не залишається як лише позбутися цих записів.

Один з найефективніших методів для відновлення втрачених даних є такий інструмент статистики як регресія. Вона являє собою статичний процес для встановлення зв‘язків між змінними, що може моделювати відношення між залежними та незалежними змінними. Регресія може бути використана майже при будь яких втратах даних, адже містить набір підходів, що базуються на різних математичних моделях, і які можуть бути підібрані для різних умов роботи з даними.

Розглянемо основні види регресії для очистки даних:[12]

1. Лінійна регресія

Цей тип регресії застосовується для моделювання відношення між двома змінними *x* та *y,* де *x* – незалежна скалярна величина , *y* – залежна величина, значення якої визначається за x. Припустимо, існує *n* точок (x_i,y_i), де *i* = 1..n. Необхідно сформувати модель, що якнайточніше зможе описати зв‘язок між x та y. З одного боку – це пряма лінія f(x) =b + a*x. З іншого – це лінія, що мінімізує суму квадратичних залишків помилок лінійної регресійної моделі. Така модуль може бути виражена через найбільші відхилення значень: err= $\sum(d_i)^2 = (y_1 - f(x_1))^2 + \ldots\ldots + (y_i - f(x_i))^2$

При підставленні першого виразу у другий отримаємо наступний вираз:

$$\text{err} = \sum(d_i)^2 = \sum_{i=1}^{n}(y_i - (a + b * x))^2 \qquad (1.3)$$

В результаті перетворень отримуємо наступну модель:

$$a\sum x_i^2 + b\sum x_i = \sum x_i y_i \quad (1.4)$$
$$a\sum x_i + b * n = \sum y_i$$

З її допомогою можна отримати пропущені значення датасету. Лінійна регресія дуже швидко відпрацьовує на невеликій кількості даних та при нескладній залежності між змінними.

2. Поліноміальна (квадратична) регресія

При створенні такої моделі, на площині з відміченими точками даних, проводиться крива, що залежна від позначених точок. У поліноміальній регресії деякі незалежні змінні можуть мати степені, більші за одиницю.

Розглянемо поліноміальну рівність:

$f(x)=a + b*x + c*x^2$ (1.5)

Для підбору необхідного для моделі коефіцієнту необхідно використати метод найменших квадратів, для цього у загальне рівняння для будь-якої помилки

$$err = \sum(d_i)^2 = (y_1 - f(x_1))^2 + .. + ((y_i - f(x_i))^2 \quad (1.6)$$

підставимо поліноміальне:

$$err = \sum(d_i)^2 = \sum_{i=1}^{n}(a + b * x + c * x^2)\ , \quad (1.7)$$

де n – кількість точок у даних, i – поточна точка, що обробляється. Таким чином, за допомогою рівняння ми підбираємо квадратичну модель типу $y = a+ b*x + c*x^2$, що має найменшу відстань від даних точок у даних.

На відміну від лінійної, квадратична регресія є більш гнучкою, та дозволяє моделювати складні взаємозв‘язки, а також її можно застосовувати у роботі з нелінійно розподіленими даними. Крім того, за допомогою вибору степені у моделі, здійснюється повний контроль над моделюванням зміних об‘єктів. Проте, даний метод має і свої недоліки. Створення поліноміальної моделі потребує ретельної уважності, мати уявлення про формат даних для

вибору найбільш підходящого параметру та степенів. При допущенні помилки у виборі степеня, модель може стати перенасиченою однотипними даними.

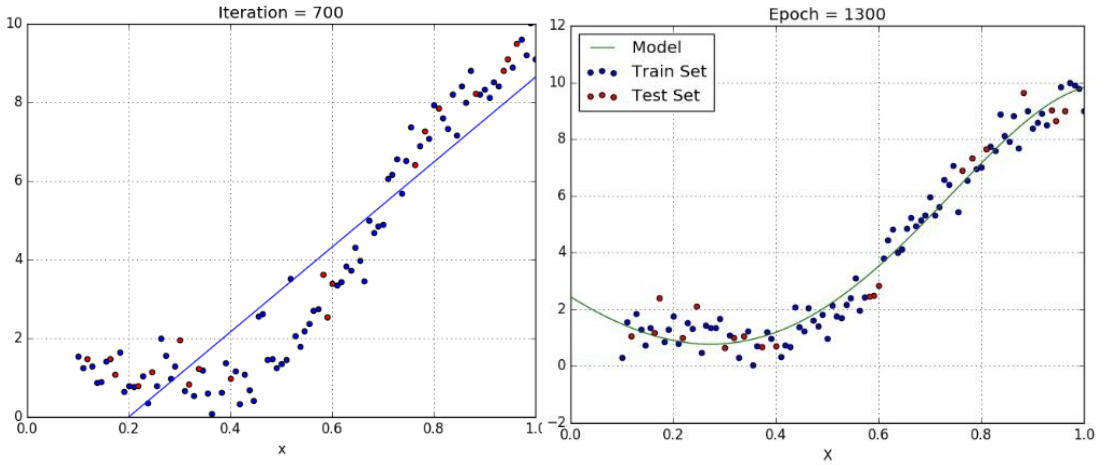

Рис 1.3. Порівняння лінійної регресії (зліва) та поліноміальної (справа) з нелінійно розподіленими даними

3. Рідж або гребінева регресія

У випадку, коли рівень колінеарності змінних - відношення незалежних змінних, близьке до лінійного - висока стандартна лінійна і поліноміальна регресії стають практично неефективними. Таким чином, через високу кореляцію змінних, кінцева регресійна модель зведена до мінімальних меж наближеного значення, тобто вона має високу дисперсією.

Наявність високого рівня колінеарності можна визначити декількома методами:

- Коефіцієнт регресії не має впливу, незважаючи на те, що, теоретично, змінна повинна мати високу кореляцію з Y.
- При додаванні або видаленні змінної з набору даних, коефіцієнт регресії сильно змінюється.
- Змінні мають високі попарні кореляції (подивіться кореляційну матрицю).

Гребінева регресія - це коригувальна міра для зниження колінеарності серед змінних, що будуть відовлюватись, у регресійній моделі. Даний метод має на меті додавання незначного фактору квадратичного зміщення для зменшення дисперсії:

$$\min \| Xw - y \|^2 + z \| w \|^2,$$

де X - це матриця змінних, w - ваги, y - достовірні дані.

Допущення даної регресії такі ж, як і в методі найменших квадратів, крім того факту, що нормальний розподіл в гребеневій регресії не передбачається.

4. Регресія методом «лассо»

У регресії лассо (LASSO, Least Absolute Shrinkage and Selection Operator), як і в гребіневій, ми додаємо умову зсуву до функції для того, щоб зменшити колінеарність, а, отже, і дисперсію моделі. Але замість квадратичного зміщення, ми використовуємо зміщення абсолютного значення:

$$\min \| Xw - y \|^2 + z \| w \|$$

5. Регресія «еластична мережа»

Еластична мережа - це гібрид методів регресії ласо і гребеневої регресії, в якому враховано ефективність обох методів.

$$\min \| Xw - y \|^2 + z\_1 \| w \| + Z\_2 \| w \|^2$$

Практичною перевагою використання регресії ласо і гребеневої регресії є те, що це дозволяє еластичній мережі успадковувати певну стабільність гребеневої регресії при обертанні. Крім того, вона створює умови для групового ефекту при високій кореляції змінних, а не обнуляє деякі з них, як метод ласо. Також регресія еластичної мережі не має обмежень за кількістю обраних змінних.

Відновлення пропущених значень є надзвичайно ефективним підходом у покращенні якості вихідного датасету, проте не враховує ряд факторів, що також не є сприятливими для кінцевого результату обробки. Після відновлення втрачених записів даним методом, неможливо передбачити чи будуть створені дублікати інших значень, і як вони вплинуть на аналітичний продукт. Окрім то-

го, передбачення зниклих рядків не може встановити стовідсоткову гарантію, що саме це значення приймав параметр під час збору інформації, або саме така характеристика була встановлена у ситуації, що спостерігалася. Слід також зазначити, що цей метод можливо застосувати лише для числових значень: у контексті роботи з текстовими даними (наприклад, нейролінгвістичним програмуванням, або парсингом веб-сторінок) даний підхід не має змісту. У подібних ситуаціях, коли відновлення даних неможливо, є сенс їх видалення для зменшення програмного навантаження.

1.3.3. Пошук та видалення записів-дублікатів [3]

Причиною такого явища як дублювання записів у даних можуть бути різноманітними, як збір даних з декількох джерел, випадкові подвійні записи до БД (наприклад, користувач заповнив форму даними та натиснув кнопку «відправити» декілька разів через довгу затримку програми або проблеми з підключенням до інтернету).

Видалення дублюючих записів у наборах даних не є само по собі надскладною операцією. Проте визначення дублікатів робить цю проблему досить комплексною. На сьогоднішній день існує ряд методів визначення дублікатів даних у датасетах, окремо для числових та текстових значень.

Один із найефективніших методів очистки числових значень, що є ефективним для великих масивів даних, є метод із використанням фільтру Блума[15] – структура даних, що дозволяє перевіряти приналежність елемента до множини. Основна ідея полягає у таких основних кроках:

1. Створити масив бітів *latexm.*

2. Вибрати *latexm* незалежних хеш-функцій $latexh_i(x)$ із областю визначення *latex[0..m−1]*

3. Для кожного елемента масиву розрахувати хеш та перетворити у біт.

4. При запиті на дублікати, застосувати хеші та перевірити, чи усі відповідні біти «увімкнені».

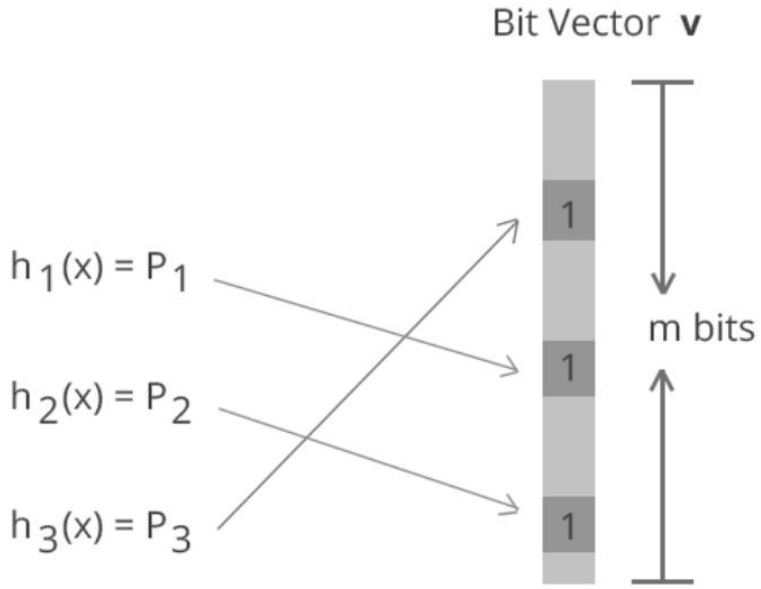

Рис 1.4. Фільтр Блума для алгоритму визначення дублікатів

Описаний у джерелі[3] метод дозволяє не просто позбуватись повторюваних текстових значень, що співпадають на 100 відсотків, а також враховувати пошкодження інформації під час збирання або передачі, випадкові одруківки, невірно зібрані дані тощо. Аби забезпечити захищенність даних, що випадково схожі на продубльовані, але є повністю правдивими, алгоритм враховує приналежність запису до певної предметної області і може вирізнити такі дані як «чисті». Даний алгоритм використовує обробку даних на основі вікон. Основні кроки алгоритму:

1. Визначити унікальність кожного атрибута набору даних (числове значення, яке повертає число унікальних даних атрибута)

2. Провести сортування за найменш унікальним атрибутом, далі відсортуйте набір даних, використовуючи кожен атрибут. Якщо така ознака не може бути визначена, використати при сортуванні набору даних будь-який інший ат-

рибут, який із більшою вірогідністю має у своїх записах дублікати, ніж інші атрибути.

3. Встановити вікно довжиною n.

4. Виконати найбільш оптимальну та швидку операцію підбору рядків між новим (n-м записом у вікні) записом і тими, що залишилися (n-1) записи в вікні попарно. Якщо кількість рядків, що співпали, нижче порогового значення, що було задано експертом, продовжити перебирати рядки за кроком 4.

5. Виконати операцію підбору рядків тільки для тих записів, співпадіння з якими вище порогу, зазначеного на початку.

6. Якщо кількість рядків зі співпадіннями між двома записами на кроці 5 вище заданого порогу і будь-який з них вже існує в наборі з дублікатами записів, використати правило транзитивності. Якщо жоден з них не існує в жодному з наборів, створити новий набір дублікатів записів з цими двома записами.

7. Якщо останній запис з будь-якого набору дублікатів записів виходить за діапазон вікна, зафіксувати знайдені до цього моменту записи дублікатів.

8. Повторювати кроки з 4 по 7 доки останній запис набору даних не буде включений у діапазон вікна та оброблений на кроці 4.

9. У отриманому наборі дублікатів залишити лише по одному запису на набір, та видалити інші.

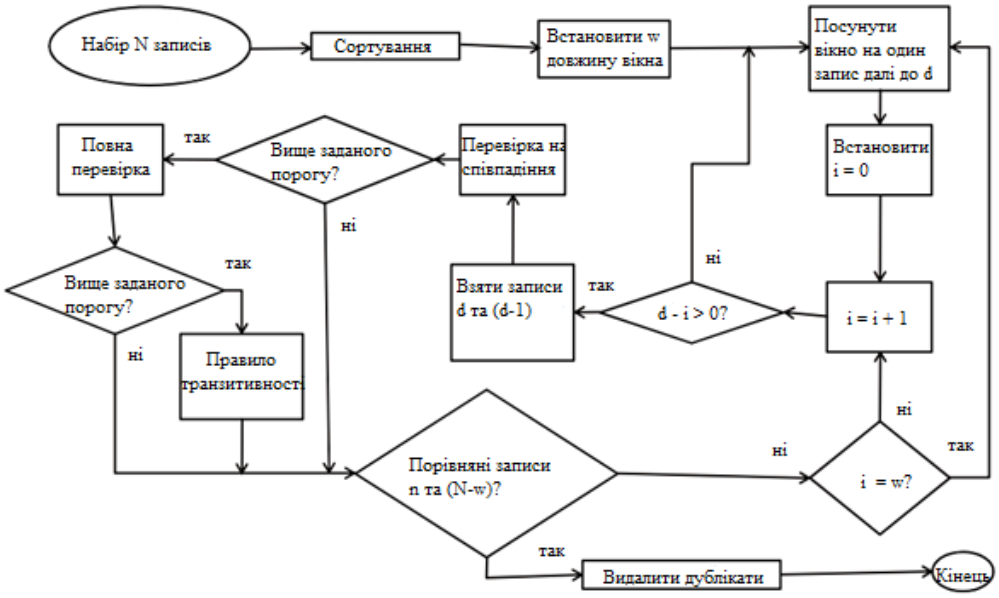

Рис 1.5. Діаграма, що описує алгоритм пошуку записів-дублікатів

При роботі з великими даними питання чи потрібно видаляти записи-дублікати поки що не має однозначної відповіді, не може встановити її програмно та залишається відкритим. Майже неможливо на сьогоднішній день за допомогою алгоритмів охарактеризувати тип та природу даних, тобто чи можливі в датасеті повтори. Наприклад, якщо відбувається збір даних про погоду, то одна й та сама температура при такому ж самому тиску може втримуватися протягом тривалого часу. У виборці, наприклад, пацієнтів з медичного закладу навпаки - дуже мала вірогідність повтору повного ім‘я, прізвища та віку у іншої людини, а навіть якщо це були дійсно різні персони, видалення одного реального запису майже не вплине на остаточний результат аналізу даних.

Отже, прийняття рішення щодо видалення повторюваних записів вирішується експертом на базі певних факторів:

- природа даних (чи є кожен запис незалежним від інших);
- мета обробки даних (моделювання, загальний аналіз);
- можливий ступінь впливу дублікатів на фінальний результат;

Що стосується недоліків даного методу, є декілька потенціальних випадків, в яких вищенаведений алгоритм є малоефективним. У ситуації зі значними пошкодженнями даних, в якій частина значень спотворюється, або губиться і стає незаповненою, значна частка датасету може бути розпізнана як дублікати та видалена. Окрім того, вищенаведений алгоритм нездатний розпізнати невалідні, непередбачені предметною областю дані, що значно вплине на подальшу обробку та результат.

1.3.3. Очищення нерелевантних записів [13]

Нерелевантними спостереженнями називають значення, що значущо відрізняються від усіх інших. Визначення таких записів визначається за допомогою міжквартильного діапазону (IQR) – даному переліку значень, на яких відбувається сортування за певним параметром та розбивка на чотири рівні частини. Діапазон між кінцем першого Q1 та початком Q3 квартилів містить у собі медіану всіх значень.

Пікові нерелевантні значення вважаються тими, що є надзвичайно низькими, або ті, що знаходяться на шкалі на відстані 1,5IQR від квартилів Q1 та Q3. Основна складність роботи з подібними значеннями – аналіз та встановлення, чи молива ситуація з таким показником насправді, чи отриманий результат є нічим іншим як програмний або апаратний збій.

Також до нерелевантних спостережень відносять ті, які насправді не відповідають конкретній проблемі, яка буде розв‘язуватись. Наприклад, якщо побудована модель створена лише для односімейних будинків, спостереження за квартирами не є значущим. Основна мета полягає у перегляді діаграм із дослідницького аналізу, схем розподілу категоричних ознак та очищення усіх сутностей та властивостей, що не належать до поточної проблеми. Це зможе зберегти дуже велику кількість часу при обробці датасету та підвищити точність, адже непотрібні поля не будуть впливати на очистку. Проте цей фактор більше стосується людського фактору та роботи експерта.

1.4. Аналіз підходів щодо покращення роботи з даними

Підготовка даних до подальшої обробки має на меті не лише їх очищення від нерелевантних, зайвих або пустих записів, а ще й їх форматування для подальших простіших та оптимальніших операцій обробки з ними - приведення до єдиного виду, стандартизація, нормалізація.

Нормалізація - це процедура попередньої обробки даних, при якій значення ознак у вхідному векторі приводяться до деякого заданому діапазону, наприклад, [0 ... 1] або [-1 ... 1]. Будучи різними за фізичним змістом, дані сильно розрізняються між собою за абсолютними величинами. Вихідні значення ознак можуть змінюватися в дуже великому діапазоні і відрізнятися один від одного на кілька порядків, або мати різну розмірність. Наприклад, значення об‘єкту «тиск» може вимірюватись як у паскалях, так і в барах, і такі ненормалізовані показники будуть чинити фатальний сплив на кінцевий результат обробки датасету.

Робота з такими показниками напряму виявиться некоректною: дисбаланс між значеннями ознак може викликати нестійкість роботи моделі, погіршити результати навчання і уповільнити процес моделювання. Після нормалізації все числові значення вхідних ознак будуть приведені до однакової області їх зміни - деякого вузького діапазону. Це дозволить звести їх разом в одній моделі і забезпечити тим самим коректну роботу обчислювальних алгоритмів.

На практиці найбільш поширені такі методи нормалізації ознак:

- Мінімакс - лінійне перетворення даних в діапазоні [0..1], де мінімальне і максимальне масштабовані значення відповідають 0 і 1 відповідно;
- Z-масштабування даних на основі середнього значення і стандартного відхилення: розподіл різниці між змінною і середнім значенням на стандартне відхилення;
- десяткове масштабування шляхом видалення десяткового роздільника значення змінної.

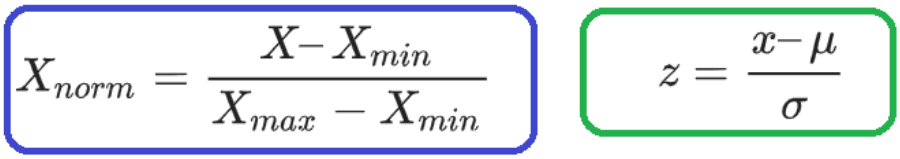

Рис 1.6. Формули нормалізації даних по методам мінімакс і Z-масштабування

На практиці минимакс і Z-масштабування мають схожі області застосовності і часто взаємозамінні. Однак, при обчисленні відстаней між точками або векторами в більшості випадків використовується Z-масштабування. А минимакс корисний для візуалізації, наприклад, щоб перенести ознаки, які кодують колір пікселя, в діапазон [0..255].

Очевидно, що вищезгадані методи підготовки даних до обробки стосуються лише числових даних. Якщо очікується робота з текстовими даними, наприклад, для нейролінгвістичного програмування (обробки природного тексту), є кілька підходів що допоможуть значно оптимізувати майбутній процес.

Набір засобів очистки тексту передбачає:

1. *Означення та видалення часток та беззмістовних слів.* Слова, що не несуть окремого змістовного значення та/або слова, що ігноруються пошуковими системами. Наприклад, для англійскої мови це можуть бути слова ‘a’, ‘an’, ‘in’, для української – ‘ті’, ‘між’, ‘до’ та ін. У середньому, у пошукових запитах англійською мовою ігнорується приблизно 130 слів.

2. *Видалення пунктуації.* Сучасні системи розпізнавання та роботи з тексти нажаль, не можуть розпізнавати змістовне значення знаків пунктуації на сьогоднішній день. Для зниження загального шуму датасету, будь-які символи пунктуації мають бути видалені. Якщо датасет має бути експортований у форматі .csv, то наявність ком порушить загальну структуру файлу, и дані не будуть валідними.

3. *Перетворення тексту до нижнього регістру.* Даний крок форматує дані до єдиного формату, та унеможливлює у подальшому створення повторюваних копій одних і тих самих слів. Наприклад, слова «Яблуко» та «яблуко» можуть бути розпізнані як різні. Завдяки перетворенню слів до єдиного регістру, такий випадок стає мало можливим.

4. *Лемматизація.* Даний пункт майже випливає із попереднього. Під лемматизацією мається на увазі перетворення слів до їх канонічного вигляду: форма інфінітиву, називний відмінок, однина). Слова у різних відмінках не змінюють свого змістовного значення, а алгоритми обробки природного тексту на даний момент не здатні розпізнавати відтінки слів, виражені за допомогою різних відмінків.

5. *Стемматизизація.* Споріднений до попереднього пункт, що має на меті видалення словобудуючих елементів, що майже не впливають на основне значення слів, проте значно зменшують їх загальну кількість, а отже і навантаження при подальшій обробці.

Підготовка тексту за цими кроками є базовим алгоритмом нормалізації слів. Після виконання цих кроків, дані готові для подальшої роботи, проте багато алгоритмів мають ще кілька етапів попередньої обробки. Один із найпоширеніших принципів – *векторизація*, перетворення текстових даних на числові для аналізу. Для виконання такої трансформації використовуються спеціальні моделі, найбільш поширеними є:

- “*Торба слів*” *(bag of words)* – детальна репрезентативна модель для спрощення обробки текстового вмісту. Вона не враховує граматику або порядок слів і потрібна, тобто попередні кроки очистки не вплинуть на кінцеву якість моделі. Головна ідея - визначення кількості входжень окремих слів в аналізований текст . На практиці “bag of words” реалізується в такий спосіб: створюється вектор довжиною в словник, для кожного слова вважається кількість входжень в текст і це число підставляється на відповідну позицію. Проте, при цьому втра-

чається порядок слів у тексті, а значить, після векторизації пропозиції, наприклад, «i have no cats» і «no, i have cats» будуть ідентичні, але протилежні за змістом. Для вирішення цієї проблеми при токенізації використовуються n-грами.

- "*n-грами*" - комбінації з n послідовних термінів для легкого розпізнавання текстового змісту. Ця модель визначає і зберігає суміжні послідовності слів в тексті . При цьому можна генерувати n-грами з букв, наприклад, щоб врахувати схожість родинних слів або помилок.
- *Word2Vec* - набір моделей для аналізу природних мов на основі дистрибутивної семантики і векторному поданні слів. Цей метод розроблений групою дослідників Google в 2013 році. Спочатку створюється словник, що навчається на вхідних текстових даних, а потім обчислюється векторне подання слів, засноване на контекстної близькості. При цьому слова, що зустрічаються в тексті поруч, у векторному поданні матимуть близькі числові координати. Отримані вектори-слова використовуються для обробки природної мови та машинного навчання.

На основі цих моделей існують і інші, більш складні, методи векторизації текстів. Практично всі ці способи реалізовані в спеціальних середовищах, наприклад, GATE, KNIME, Orange, RapidMiner, LPU, а також спеціальних бібліотеках на мовах програмування Python і R. Даний підхід не є універсальним, проте підходить для більшості задач аналізу текстів, наприклад, визначення загального настрою у відгуках про продукт або компанію. [14]

Висновки:

1. Проаналізовано існуючі методи попередньої обробки даних у системах обробки великих даних за рахунок аналізу сучасних досліджень у цій сфері, визначено потенційні проблеми вихідних даних та причини, що можуть призводити до них.
2. Сформульовано переваги та недоліки кожного з проаналізованих методів очистки даних, які показали, що для покращення якості вхідного потоку

даних необхідно використовувати певний метод в залежності від особливостей певного потоку даних

3. Проведено аналіз методів щодо загального покращення якості даних та підготовки їх до подальшої ефективної обробки. Визначено універсальний підхід щодо загального покращення якості вхідних даних за рахунок приведення їх до єдиної структури.

РОЗДІЛ 2.
ВДОСКОНАЛЕНИЙ ПІДХІД ЩОДО ПОПЕРЕДНЬОЇ ОБРОБКИ ДАНИХ

2.1. Модифікований метод очистки даних

Усі найпоширеніші методи попередньої обробки у своїй суті націлені на виправлення лише одного аспекту забруднення даних. Основною ідеєю модифікованого алгоритму очищення датасету є додавання додаткового кроку у базовий процес, мета якого - поєднання різних методів у єдиний алгоритм обробки.

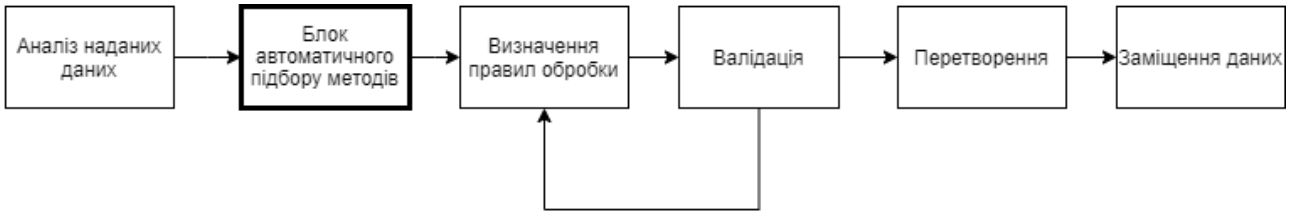

Рис 2.1. Блок-схема модифікованого процесу очистки даних

Ще раз перелічимо аспекти, що будуть застосовуватись у процесі очистки даних:

1. Відновлення пропущених та пошкоджених записів
2. Видалення дублюючих записів
3. Виправлення пікових та нерелевантних значень

Детально роботу алгоритмів, що націлені на роботу даними аспектами описано у першому розділі.

Проте не всі дані можуть бути зібрані з різних джерел, а отже містити різні структури моделей та мати різні зв‘язки між параметрами, або містити пропущені дані, бо датчики збору інформації відпрацювали високоякісно протягом всього періоду роботи. Або можуть бути відсутні дублікати, адже сховище даних, з якого створювався набір, не містить у собі повторюваних записів, або вони могли бути видалені вручну адміністратором бази даних. Отже, існуючі ме-

тоди попередньої обробки великих даних не є «срібною пулею» - для кожного окремого випадку та кожного набору даних, що йде на обробку, має бути встановлений відповідний процес очистки.

Запропонований модифікований алгоритм має на меті проаналізувати вхідні дані та адаптивно створити унікальний порядок дій щодо очистки. Для вибору алгоритмів для обробки мають бути сформовані чіткі критерії вибору. Тобто, на меті стоїть точно сформулювати при яких умовах той чи інший метод попередньої обробки доцільно застосувати. Таким чином вдасться розробити алгоритм, що динамічно буде здатний скласти порядок та складові обробки, і тим самим не лише провести ефективну очистку даних, а й знизити навантаження на систему та кількість часових та програмних ресурсів.

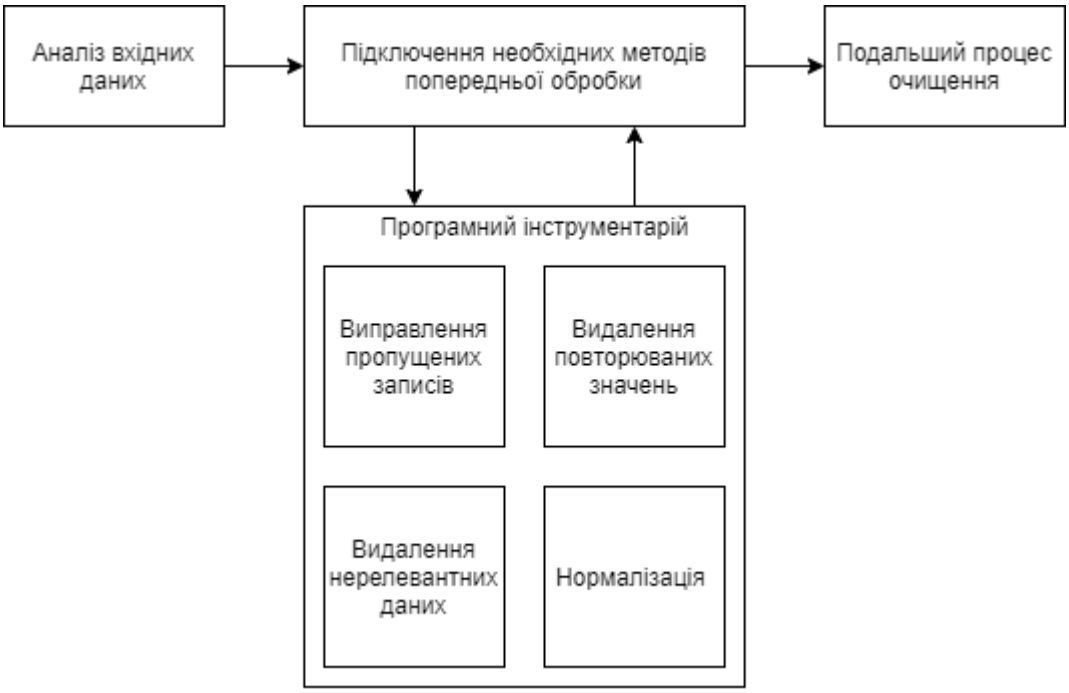

Рис 2.2. Структурна схема інструментальних засобів очищення даних за модифікованим алгоритмом

2.2. Формування критеріїв вибору методу очистки даних

Для частини методів очистки неможливо сказати однозначно чи доречно використати даний метод через неоднозначність вибору та варіативності кількості даних – при якій кількості відповідних випадків необхідно застосовувати даний підхід. До прикладу, у датасеті 1% пошкоджених даних, що необхідно виправити. Якщо набір містить мільйон записів, то 1% від усього числа складуть десять тисяч, і це може значно вплинути на результат і їх виправлення є доцільним. Якщо ж це датасет зібраний з одного з десятків джерел з невеликої установи і містить у собі тисячу записів, то 1% складатиме лише 10 записів, що значно не зможе вплинути на кінцевий результат, особливо, якщо датасет доповнять вже очищені дані з інших джерел. Проте ресурсів буде використано майже еквівалентно першому випадку.

Більш того, неможливо програмно встановити точність очистки, на яку розраховує експерт, адже чим вища точність, тим більше часу необхідно до забезпечення заданого відсотку якості. Тому у даному методі пропонується ввести параметр, що має бути встановлений експертом, а саме максимально допустимий відсоток забруднення z_0.

Наступним кроком має бути проведений аналіз, на основі якого буде проводитись підбір алгоритмів для очистки. На основі сформованих метаданих буде проведена валідація за критеріями підбору кожного методу, встановлений відсоток даних, що підлягає очистці z. За умови $z_0 <= z$, алгоритм буде включено до виконання програми очистки датасету.

Розглянемо випадки, коли застосування методу очистки є доцільним:

1. Відновлення пропущених записів

Найпростішим випадком є визначення пропущених записів. До записів, що потребують очистки, будуть віднесені записи зі значеннями, що містять NULL, але за визначенням, що базується на метаданих, таких значень набувати не можуть. Записи, всі значення яких рівні NULL очистці, адже такий випадок

дуже схожий на повну втрату інформації у саме цей момент часу. Більш того, такі записи не мають жодного впливу на кінцевий результат обробки.

2. Видалення дублюючих записів

Перш за все, підключення до основного алгоритму даного методу, ми будемо базуватись на метаданих, що були надані експертом, а саме на наступних факторах:

- Якщо дублікати за певним параметром заборонені, відбуватиметься пошук та їх видалення. Також це стосується наявність дублікатів-об‘єктів загалом, або повторюваних записів
- Перевірка дублікатів за первинним ключем (якщо у метаданих він наданий). Спершу перевіряються дані за зазначеними полями у ключі, а потім – усі записи без них, тобто, якщо ми вилучимо параметри ключа, чи не буде у записах повністю ідентичних даних.

Якщо алгоритм все ж буде підключений до загального процесу попередньої обробки, то будемо орієнтуватись на параметр $z_{0,}$ що встановлений експертом, де z – загальна кількість знайдених дублікатів. За умови $z_0 >= z$, рівень забруднення значеннями дублікатами є допустимим. При умові $z_0 <= z$ вплив хибних значень може бути суттєвим, тому очистка від повторюваних записів буде проведений.

3. Виправлення пікових та нерелевантних значень

Даний метод доцільно застосовувати, коли експертом були надані метадані, що описують дані за наступними характеристиками: який тип даних вони приймають, чи залежний параметр від іншого або навпаки.

Для числових - у якій розмірності надаються дані, чи можуть мати негативні значення, значення нуля; які мають верхнє та нижнє порогові значення, чи мають певну структуру чи вид, наприклад, можуть приймати лише значення степені двійки.

Для текстових – чи мають параметри маску-шаблон для електронної адреси або телефону; чи представляють собою інший тип даних, наприклад, дата; якщо так, то чи повинні бути сконвертовані у інший тип; чи можуть містити спеціальні символи, пропуски; чи містять певні слова, що мають бути видалені.

2.3 Формування критеріїв якості даних

Метрики якості даних (DQ Metrics) – це основний потік, в якому безпосередньо відбуваються функції оцінки якості та очистки даних, фундамент, на якому будується оцінка очищення даних. Кожна метрика має формат типу «питання-відповідь», де кожен запит має пов‘язану відповідь. Наприклад, загальна кількість пропущених значень за параметром X є DQ метрикою, значення якої впливає на показник якості даних і формуванні алгоритму очистки: заповнення відсутніх параметрів, видалення дублікатів тощо. [16]

Наприклад, кількість пропущених значень у стовпці X є показником DQ. Значення цього показника допомагає оцінити якість даних і відповідно будує дію очищення, подібно до заміни всіх відсутніх значень у X певним значенням. У алгоритмі, що пропонується у роботі, таким показником буде параметр z_0.

Для підвищення ефективності попередньої обробки даних методи очистки можуть бути запущені у повторюваному циклі до тих пір, поки необхідного рівня якості датасету, встановленого експертом, не буде досягнуто.

Ефективність запропонованого підходу щодо очистки даних на практиці буде продемонстрована на прикладі кластеризації великих даних, а саме – застосування методу K-середніх.

Кластеризація - це поділ множини вхідних векторів на групи (кластери) за ступенем «схожості» один на одного. Кластеризація в Data Science набуває цінності, коли виступає одним з етапів аналізу даних, побудови закінченого аналітичного рішення. Аналітику загалом легше виділити групи схожих об'єктів, вивчити їх особливості і побудувати для кожної групи окрему модель, ніж створювати одну загальну модель для всіх даних. Таким прийомом постійно корис-

туються в маркетингу, виділяючи групи клієнтів, покупців, товарів і розробляючи для кожної з них окрему стратегію.

Найпростішим методом кластеризації в класичній реалізації є алгоритм k-means (k-середніх). Він розбиває безліч елементів векторного простору на заздалегідь відоме число кластерів k. Основна ідея алгоритму полягає у тому, щоб мінімізувати середньоквадратичне відхилення на точках кожного кластера. На кожній ітерації заново обчислюють центр мас для кожного кластера, отриманого на попередньому кроці, і на їх основі вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближчим. Алгоритм завершується, коли на якийсь ітерації не відбувається зміни кластерів.

Алгоритм має певні недоліки, наприклад, необхідно знати наперед кількість кластерів, на які розбивається датасет. В реалізації цей параметр заздалегіть зазначає експерт. Далі, алгоритм дуже чутливий до вибору початкових центрів кластерів. Класичний варіант має на увазі випадковий вибір кластороров, що дуже часто було джерелом похибки. Як варіант вирішення, для даного дослідження, де робота проводиться з однією і тою самою вибіркою даних, пропонується брати перші k записів як центри початкових кластерів, де k – параметр, заданий експертом для зазначення кількості кластерів. Крім того, алгоритм не може враховувати ситуації, коли об'єкт належить до різних кластерів в рівній мірі або не належить жодному. Для даного дослідження цією похибкою можемо знехтувати.

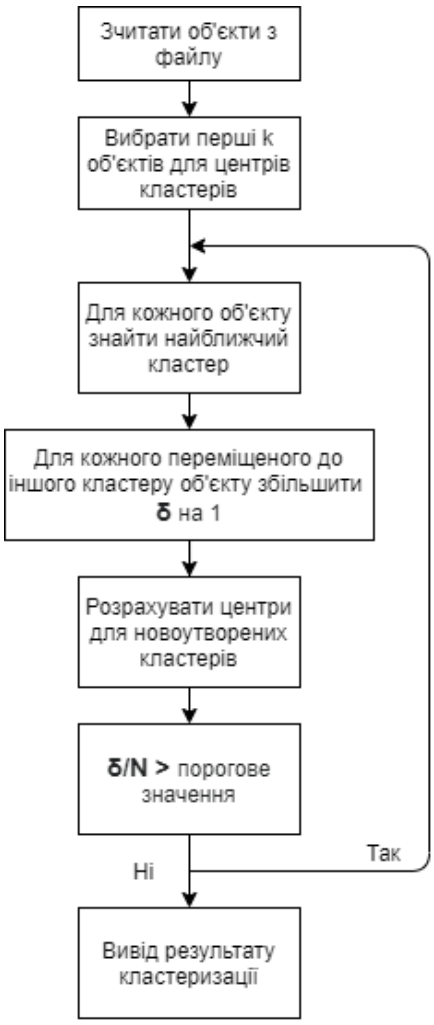

Рис 2.3. Загальна діаграма підходу алгоритму кластеризації K-середніх

Висновки:

1. Запропоновано вдосконалений метод попередньої обробки даних шляхом поєднання різних алгоритмів очистки, де будуть використовуватись переваги та усуватись недоліки кожного.

2. Запропоновано алгоритм динамічного застосування необхідних базових методів попередньої обробки даних за рахунок підключення інструментів очистки при виконанні специфічних умов.

3. Розроблено критерії для вибору потрібних методів після аналізу вхідних даних та метаданих шляхом порівняння кількості нерелевантних забруднених даних та порогового значення, заданого експертом.

РОЗДІЛ 3.
МОДЕЛЮВАННЯ ПРОТОТИПУ СИСТЕМИ ОЧИСТКИ ДАНИХ ІЗ ЗАСТОСУВАННЯМ МОДИФІКОВАНОГО МЕТОДУ

3.1. Опис загального підходу та архітектури системи

Перед початком розробки системи, реалізації були поставлені такі цілі:

- реалізувати класичний метод очистки великих даних;
- реалізувати модифікований алгоритм очистки, забезпечити можливість його універсального підключення до додатків як окремого модуля;
- продемонструвати результати обробки у проекті кластеризації великих даних;

Першим кроком до вирішення реалізації описаного методу є вибір технологій, за допомогою яких буде на практиці виконано, перевірено ефективність запропонованого алгоритму та класичних методів попередньої обробки даних. На сьогоднішній день існує безліч серверних мов програмування, що дозволяють ефективно працювати з даними та широко застосовуються у сфері Data Science для розв'язання різноманітних задач. Вибір мови для практичної реалізації зосередився на таких варіантах:

1. Python

Перший реліз мови Python відбувся під авторством Гвідо ван Россума у 1991 році. З того часу популярність мови надзвичайно зросла, і набула статусу мови програмування загального призначення, що широко застосовується в роботі з великими даними. Остання версія на сьогоднішній день - 3.8.3. До переваг данох мови можна віднести високу популярність, а отже і безліч розширень до додатків і підтримку спільноти розробників. Python має простий і зрозумілий для новачків синтаксис, що робить його відмінним кандидатом на роль першої мови програмування з низьким порогом входу. Пакети роботи з великими даними, як pandas, scikit-learn і Tensorflow роблять Python відмінним варіантом для сучасних додатків з машинним навчанням. Щодо недоліків, то Python - мова

з динамічною типізацією. Так що слід бути уважним і очікувати час від часу помилок із перетворенням даних. Крім того, за кількістю пакетів для статистичного аналізу, дана мова програє своїм конкурентам, тому не є ідеальним кандидатом у виконанні поставлених вище задач.

2. SQL

SQL (Structured Query Language) створений для визначення, управління та створення запитів до реляційних баз даних. Він з'явився в 1974 році і з тих пір зазнав безліч змін, але його основні принципи залишилися незмінними. Дана мова має ряд плюсів, як високу ефективність при роботі з реляційними базами даних і надвеликими обсягами інформації і тому має широку застосовуваність. Декларативний синтаксис робить SQL досить інтуїтивною для розуміння мовою. Проте слід зазначити, що аналітичні можливості SQL досить обмежені. Все що вам доступно - це базові функції аналізу, як підсумовування, підрахунок, виведення середнього значення тощо. Саме через це аналізувати та змінювати дані за допомогою цієї технології не буде дуже зручним.

3. MATLAB

MATLAB – продукт зі своєю мовою програмування, призначений для обчислень та моделювання, використовується в академічних колах і різних сферах промисловості. Розробка та ліцензування проводиться компанією MathWorks. Даний інструмент ідеально підходить для додатків, що вимагають складних математичних функцій, містить ряд вбудованих функцій для візуалізації даних, широко застосовується областях фізики, інженерії та прикладної математики. Проте реалізація за допомогою цього продукту не забезпечить універсального використання на практиці у різних додатках як окремого модуля.

4. JavaScript

JavaScript відомий в першу чергу як браузерна мова, проте з появою Node.js, закріпив за собою позиції і як серверна. Зявились такі пакети для роботи з даними, як brain.js і synaptic.js, проте їх кількість значно менша порівняно з

іншими мовами, отже використання JavaScript в Data Science досі обмежене, незважаючи на існування Node.js майже одинадцять років.

5. C#

C# - високопродуктивна компільована мова загального призначення, розроблена компанією Microsoft. Строга типізація анулює потенціальні помилки, повязані з перетворенням типів даних, що для додатків, які працюють з великими обсягами даних є вкрай важливим. Одні і ті же самі компоненти коду можуть бути використані і для написання бізнес-логіки, і для аналітики великих обсягів даних, на що не здатні інші мови програмування для Data Science. Вже декілька років ведеться активна розробка над набором пакетів для машинного навчання та роботи з великими даними – ML.NET. Він містить в собі інструменти для роботи з найпопулярнішими фреймворками даного напряму, такими як TensorFlow, OnnxRuntime, Mkl.Redist та рядом інших.

Отже, для реалізації алгоритму була обрана мова програмування C# з використанням платформи .NET Core – продукт, що дозволяє створювати різноманітні кросплатформенні додатки для Windows, MacOS X і Linux. Сьогодні це дуже важливо в технологічному світі, який все більше орієнтується на гнучкість і сегментування, коли справа доходить до операційних систем і платформ.

Доступність .NET Core на інших платформах, відмінних від Windows, робить його кращим кандидатом для використання багатьма розробниками, включаючи розробників Mac і Linux. .Net Core визначає вже свої додатки як програмно-орієнтовані, а не від платформо-орієнтовані [1]

Для створення додатку була обрана архітектура типу «клієнт-сервер». Розподіл функціоналу інтерфейсу користувача та бізнес-логіки дає покращення у таких аспектах як портативність інтерфейсу користувача на різноманітних платформах, покращена масштабованість за рахунок спрощення компоненти сервера.

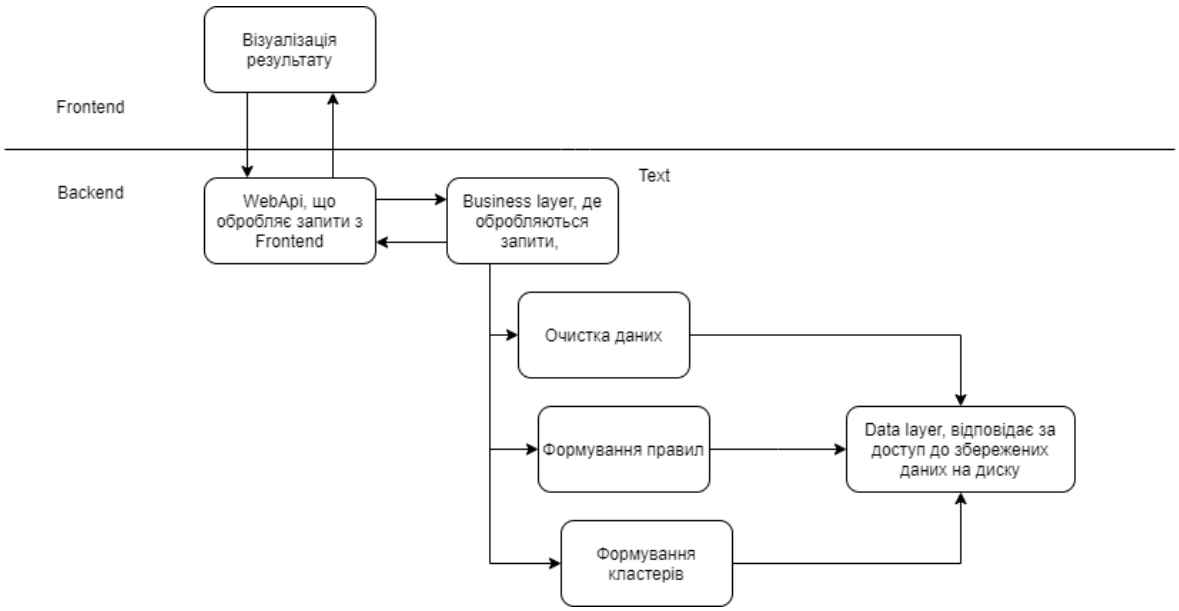

Рис 3.1. Архітектура створеного додатку

Серверна частина додатку реалізована на за допомогою WebApi .Net Core, під яким мається на увазі розширюваний фреймворк для будування сервісів на базі протоколу HTTP, що дозволяє повертати відповідь на запит до будь-якого девайсу. У даній реалізації запит буде відправлятись від клієнтської частини додатку.

Клієнтська частина веб-додатку реалізована за допомогою Angular – open source платформи від Google для розробки веб-додатків на мові Typescript. Цей фреймворк спецалізується на створенні single-page додатків, що підходить до контексту поставленої задачі. Графічний інтерфейс міститиме представлення для інтерфейсу щодо очистки даних та безпосередньо кластеризації, де можливо буде побачити результат обробки даних.

Інтерфейс, що представляє кластеризацію має в собі графічний інтерфейс для вводу даних для обробки та графіки для виводу результатів даних. Система очікує вибору файлу з даними з машини користувача та число кількості кластерів для проведення аналізу. Після повернення відповіді від серверу, сформовані кластери відображаються на графіку. Додатково можна сформувати їх проекції

на одну із осей, що використовується пізніше для подальшого формування правил бази знань. Усі графіки у додатку будуються з використанням open-source Javascript-бібліотеки Plotly.js. Написана на основі популярних бібліотек d3.js та slack.gl, це високорівнева декларативна бібліотека графіків [1]

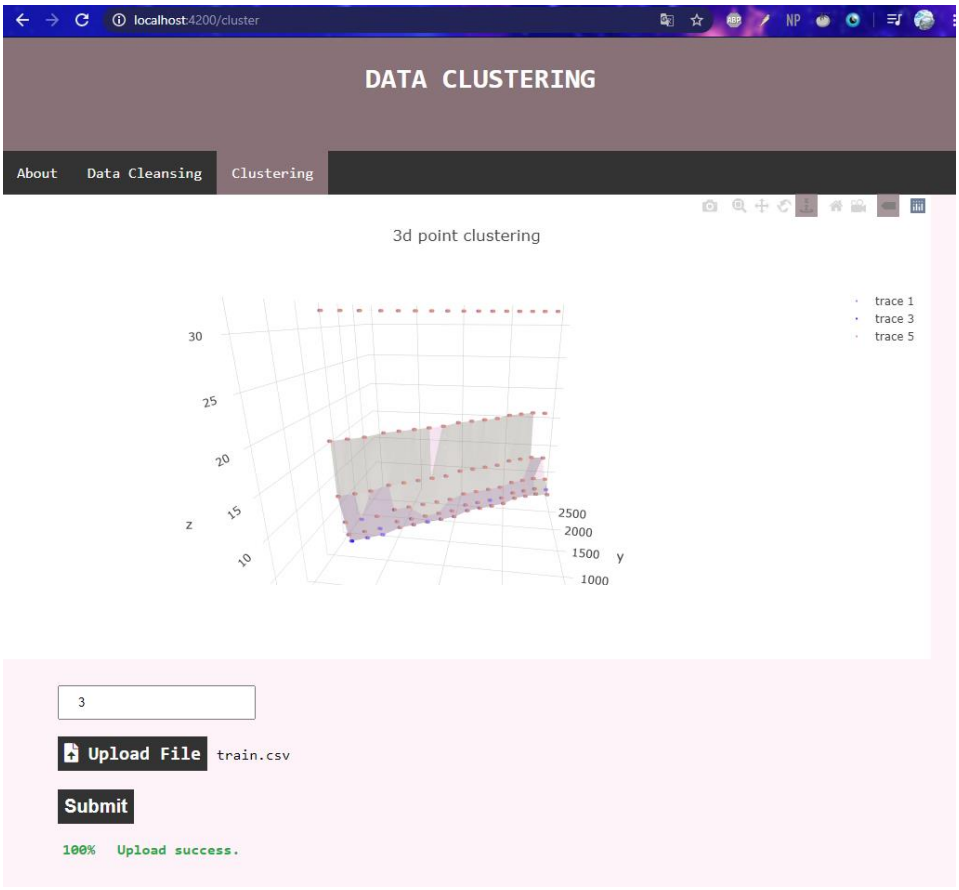

Рис. 3.2. Інтерфейс сторінки кластеризації

Сторінка, що дозволяє сформувати запит на очистку даних, очікує вибору файла з машини користувача та заповнення форми. Форма має два основні режими заповнення:

- з відміченою чекою «Auto Cleansing» - користувач обирає використання адаптивного алгоритму очистки; при виборі даного пункту, усі інші опції блокуються для вибору;

- з невідміченою чекою «Auto Cleansing» - користувач обирає власноруч налаштовувати необхідні параметри для очистки даних; необхідно додатково ввести яке порогове значення забруднення даних є допустимим;

Можливі налаштуванні при ручному налаштуванні очистки:

1. *Фільтрування даних* – видалення нерелевантних пікових значень. Після утворення схеми об’єкту даних, користувачу пропонується для кожного ввести максимальне та мінімальне значення.
2. *Відновлення пропущених значень* – заповнення відсутніх значень параметрів у наборах даних. Змінна заповнюються відповідним середнім для праметра значенням.
3. *Нормалізація даних* – приведення значень набору даних до діапазону [0 ... 1]. Якщо на етапі аналізу були виявлені від’ємні значення, діапазон розширюється до [-1 ... 1].
4. *Видалення дублікатів* – визначення повторюваних записів у датасеті.

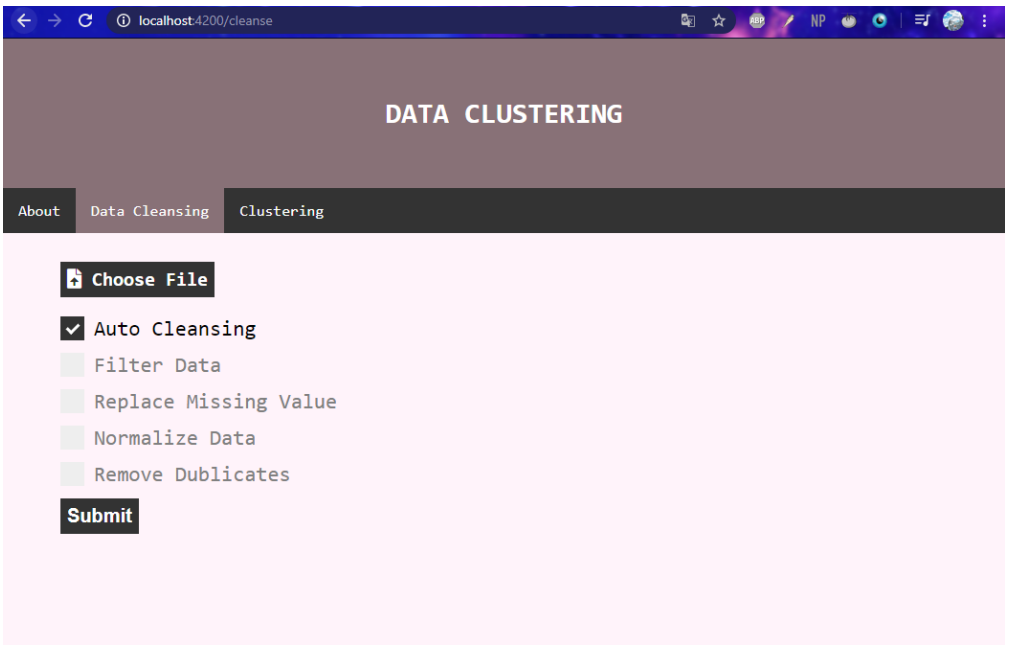

Рис 3.3. Інтерфейс очистки даних

3.2. Опис проведення дослідження

Для тестування реалізованого додатку використовувались дані щодо роботи серверних машин, що містять у собі такі показники:

- *Номер зразку* – будь-яке натуральне число;
- *Кількість потоків*;
- *Частота обробки даних* – число у вигляді 2^n, де n – натуральне число;
- *Енергія, яку споживатиме машина*;

Дослідження проводилось наступним чином: спочатку було проведено кластеризацію неочищених даних, далі – почергово підключались реалізовані в системі методи очистки, і в кінці – проведено обробку очищених за запропонованим алгоритмом дані. Кількість кластерів була задана однакова на всіх етапах, а саме вісім. На кожному етапі фіксувались графіки кластеризованих даних, їх проекція на одну з осей простору та кількість ітерацій, за які було проведено обробку. Нижче представлені отримані результати.

3.2.1. Дослідження графіку сформованих кластерів

Першим етапом експерименту було кластеризування попередньо ніяк не оброблених даних. Як можна побачити з нижченаведеного графіку, кластери практично не сформовані через неправильно визначені центри кластерів. Більшість невалідних значень приєднані до кластерів випадковим чином.

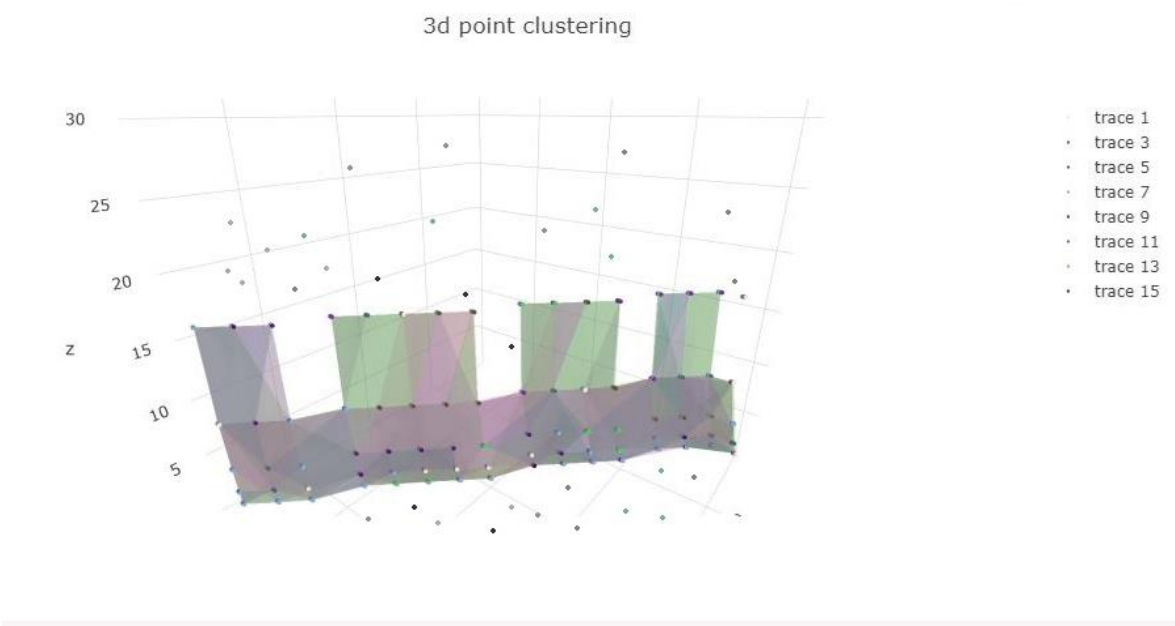

Рис 3.4. Результат кластеризації неочищених даних

Наступним етапом було проведення очищення за допомогою метода відновлення відсутніх даних. З графіку можна зробити висновок, що через відновлення алгоритмом втрачених значень, центри кластерів були встановлені більш чітко, тому обробка пройшла з вищою точністю. Присутні пікові нерелевантні значення, що порушують структури кластерів.

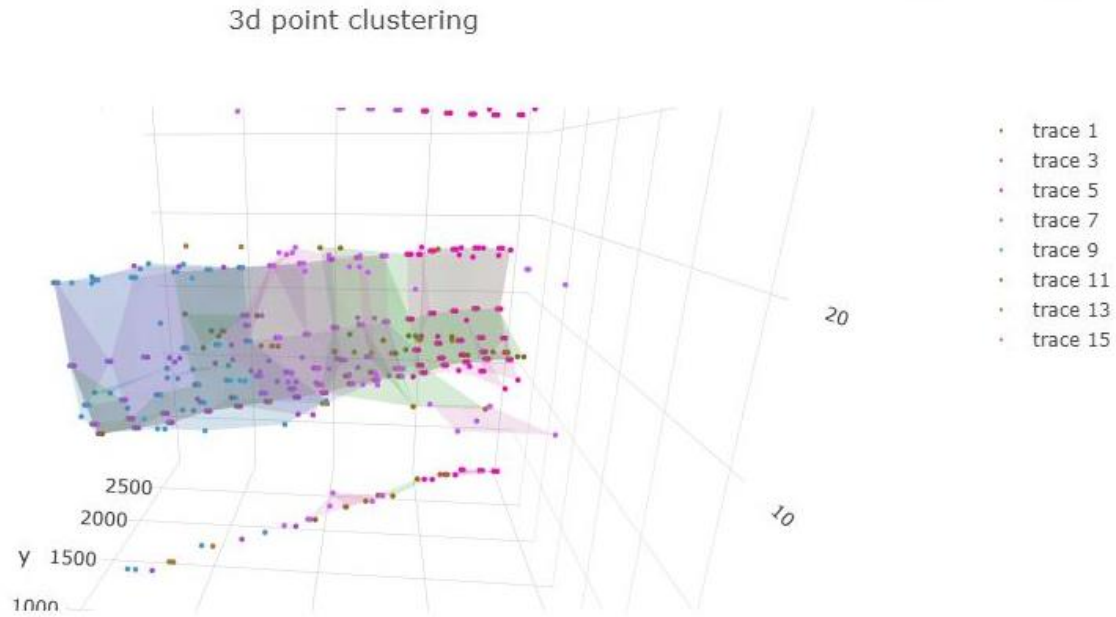

Рис 3.5. Результат кластеризації після відновлення пропущених даних

Далі було проведенно попередню обробку даних за допомгою виправлення нерелевантних записів. В результаті проведення даного етапу зникли пікові значення, що не були валідними. Помітні кластери з порушеною цілісністю, через зміщені центри кластерів.

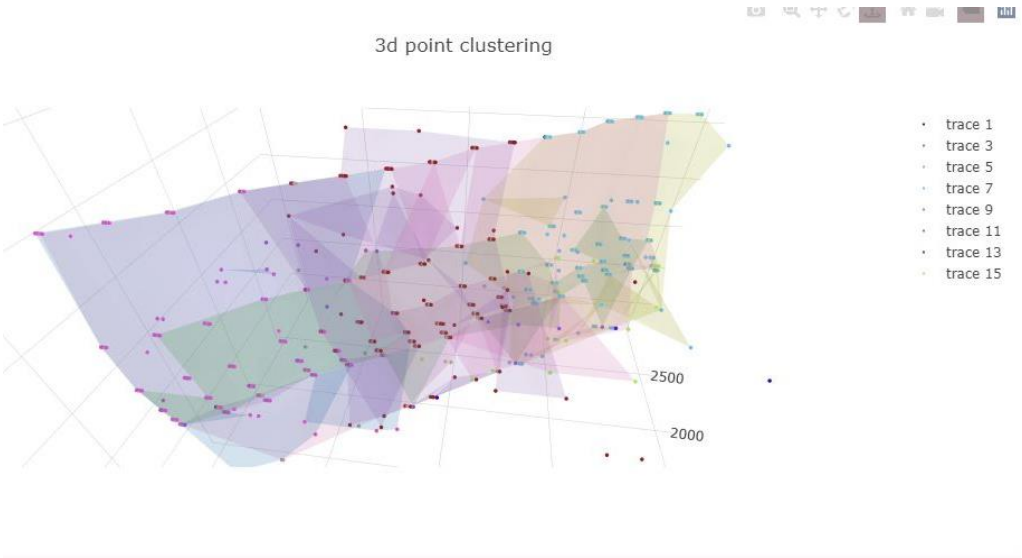

Рис 3.6. Результат кластеризації після виправлення нерелевантних значень

Після проведення очищення методом видалення повторюваних значень очевидні неправильно сформовані кластери та пікові значення, що були неправильно приєднані до кластерів

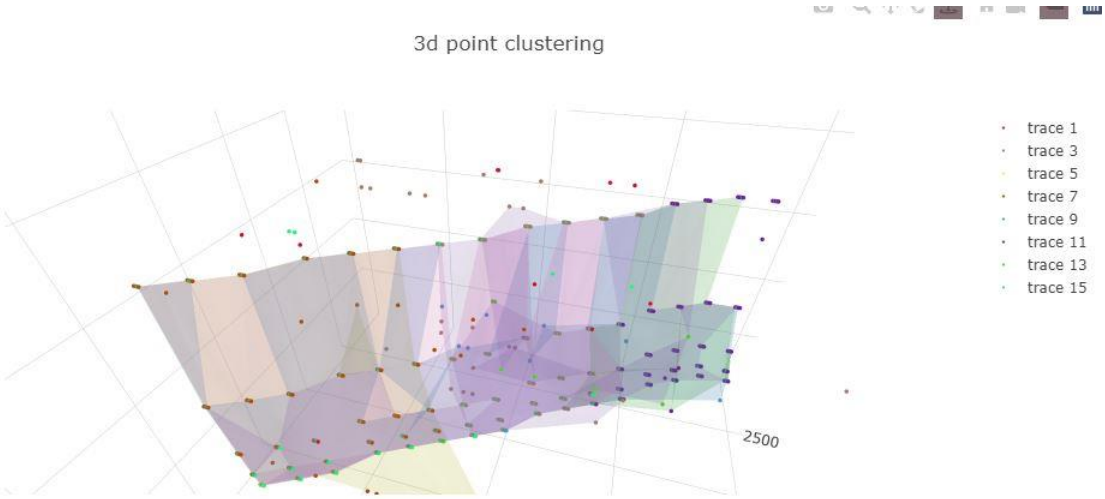

Рис. 3.7. Результат кластеризації після видалення дублікатів

Наприкінці було проведено очистку за допомогою запропонованого алгоритму. Після виконання очистки модифікованим методом кластери сформувались помітно чіткіше через правильно визначені центри. На графіку відсутні пікові значення, усі зазначені точки розподілені між кластерами рівномірно.

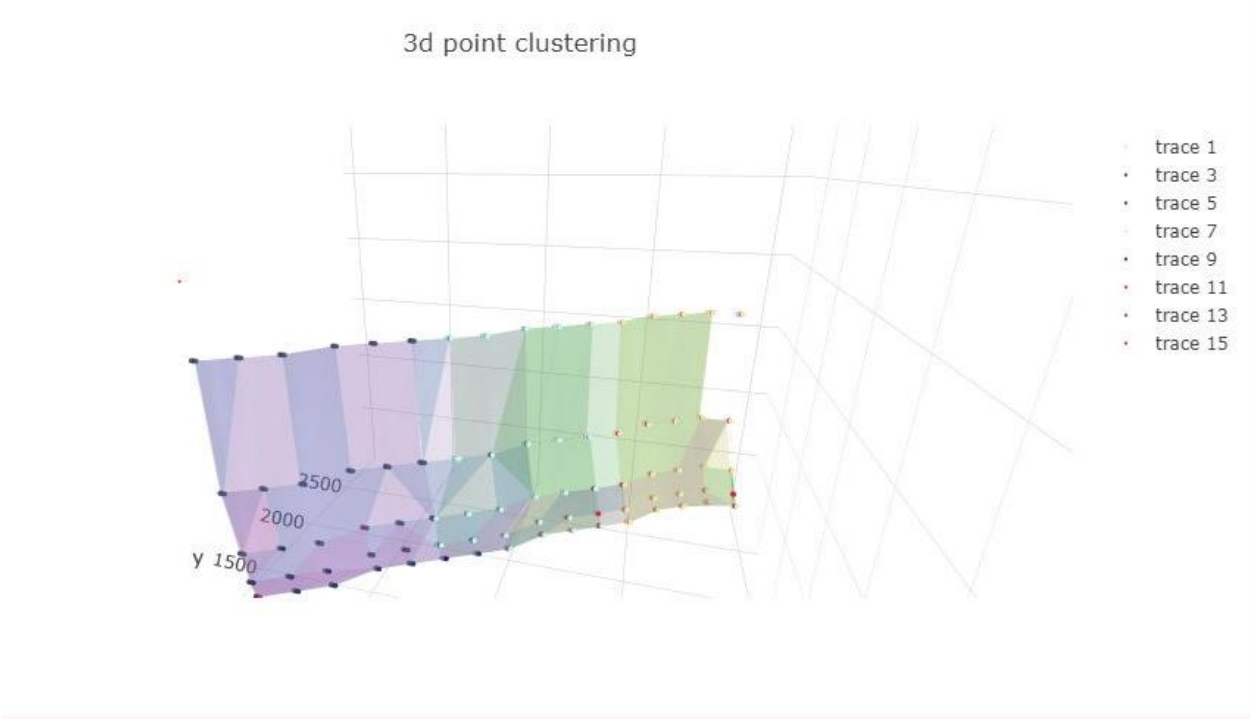

Рис 3.8. Результат кластеризації після застосування запропонованого алгоритму

Отже, запропонований алгоритм попередньої обробки великих даних значно підвищив якість результату проведення кластеризації порівняно з класичними засобами попередньої обробки даних.

3.2.2. Дослідження проекцій кластерів на вісь простору

В результаті проведення експерименту загалом було отримано два види графіків проекцій на вісь. Для спрощення роботи бізнес-логіки вісі x,y та z були позначені цілими числами 0,1 та 2 відповідно.

Рис 3.9. Графік проекції при застосуванні модифікованого алгоритму та алгоритму виправлення відсутніх значень

Рис 3.10. Графік проекції при роботі з неочищенними даними, застосуванні алгоритмів видалення пікових значень та видалення дублікатів

Як видно з наведених графіків, при наявних відсутніх значеннях деякі з кластерів при проекції на вісь можуть мати досить велику дисперсію, а отже вносити значну похибку у подальшу побудову логічних правил.

3.2.3. Дослідження кількості ітерацій

За результатами проведеного дослідження були зібрані такі результати щодо кількості ітерацій при виконанні кластеризації на кожному етапі.

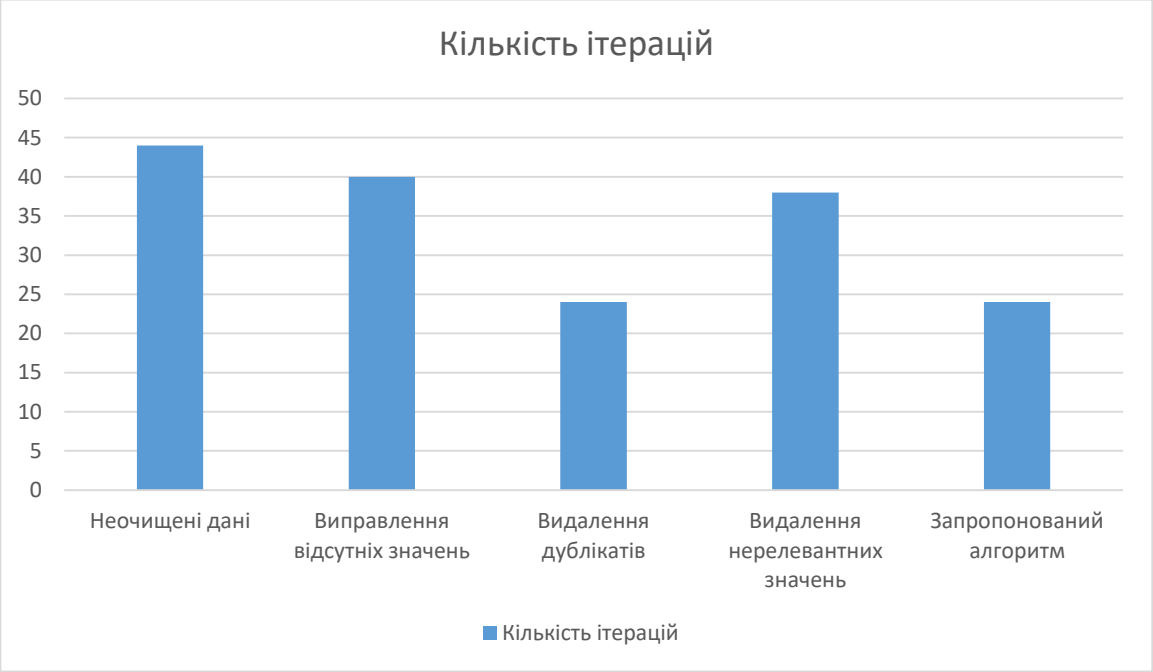

Рис 3.11. Кількість ітерацій під час кластеризації

Таким чином, застосування методу видалення дублікатів, а також його включення у запропонований алгоритм попередньої очистки даних, допомагає знизити кількість ітерацій, а отже програмних та числових ресурсів при виконанні обробки великих даних.

Висновки:

1. Проведено аналіз мов програмування, за допомогою яких можливо реалізувати запропонований алгоритм, класичні методи очистки даних та провести кластеризацію для проведення дослідження.

2. Розроблено веб-додаток згідно сформованих цілей, функціональність якого включає застосування ручного налаштування попередньої обробки даних, застосування автоматичного динамічного алгоритму та проведення після очистки процесу кластеризації.

3. Проведено модельний експеримент, на результаті якого продемонстровано ефективність розробленого методу попередньої обробки даних за критеріями якості сформованих кластерів, їх проекцій на осі простору та швидкості обробки (кількістю ітерацій аналізу).

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Проведено аналіз факторів, що впливають на якість вхідних даних, розглянуто потенційні проблеми, до яких вони можуть призводити. Проаналізовано найчастіше використовувані методи попередньої обробки інформації, для кожного сформовано свої переваги та недоліки.

2. Як науковий результат запропоновано вдосконалений метод щодо попередньої обробки даних.

3. Як практичний результат розроблено програмне забезпечення на основі запропонованого алгоритму, що дозволяє очищувати великі обсяги даних у автоматизованому режимі з підвищеною якістю кінцевих результатів.

4. Проведено порівняння класичних методів попередньої обробки даних із запропонованим. Практично продемонстровано на результаті проведеного модельного експерименту на процесі кластеризації великих даних, що запропонований метод є найбільш ефективним у трьох аспектах проведення обробки інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Buhaienko Y., Globa L.S., Liashenko A., Grebinichenko M. “Analysis of clustering algorithms for use in the universal data processing system”, OSTIS 2020

2. R. Deepa. Methods and techniques to evaluate the performance of Data Cleansing Algorithms for very Large Database Systems / R. Deepa, Dr. R Manicka Chezian. // IJARCET. - 2016

3. A Domain-Independent Data Cleaning Algorithm for Detecting Similar-Duplicates / Kazi Shah Nawaz Ripon Ashiqur Rahman and G.M. Atiqur Rahaman. // JOURNAL OF COMPUTERS. – 2010

4. A Data Cleaning Method Based on Association Rules [Електронний ресурс] / W.Weijie, Z. Mingwei, Z. Bin, T. Xiaochun – Режим доступу до ресурсу: www.atlantis-press.com.

5. Deepshikha A. An Optimum Model for the Retrieval of Missing Values for Data Cleansing using Regression Analysis / A. Deepshikha, A. V. B.. // International Journal of Computer Applications. - 2015

6. Liashenko A., Buhaienko Y. "A clustering method for processing large volumes of data". MODERN CHALLENGES IN TELECOMMUNICATIONS NTUU "KPI", Kiev 2019.

7. “Data Cleansing for Web Information Retrieval using Query Independent Features [Електронний ресурс] / L.Yiqun, Z. Min, R. Liyun, M. Shaoping – Режим доступу до ресурсу: www.thuir.cn.

8. Pehwa P. “An Efficient Algorithm for Data Cleaning [Електронний ресурс] / P. Pehwa. – 2011. – Режим доступу до ресурсу: www.igiglobal.com.

9. KavithaKumar R. “Attribute Correction-Data cleaning using Association Rule and Clustering Methods” / R. KavithaKumar, D. Chandrasekaran. // IJDKP. – 2011.

10. Qian Y. The role of visualisation in effective data cleaning / Y. Qian, Z. Kang. // Proceddins of 2005 ACM symposium on applied computing. – 2005.

11. Rajashree Y. A Review of Data Cleaning Algorithms for Data Warehouse Systems / Y. Rajashree, R. Kulkarni. // IJCSIT. – 2012.

11.www.it.ua/ru/knowledge-base/technology-innovation/big-data-bolshie-dannye

12. 5 Types of Regression and their properties [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: towardsdatascience.com.

13. The Ultimate Guide to Data Cleaning [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: towardsdatascience.com.

14. Beginner's Guide to Data Cleaning and Feature Extraction in NLP [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: towardsdatascience.com/.

15. How to remove duplicates in large datasets [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: clevertap.com/.

16. Dirty Data — Quality Assessment & Cleaning Measures [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: towardsdatascience.com.